

AD-A125 766

A VECTORIZED GENERAL SPARSITY SOLVER(U) MICHIGAN UNIV
ANN ARBOR SYSTEMS ENGINEERING LAB D A CALAHAN
01 OCT 82 SEL-168 AFOSR-TR-83-0876 AFOSR-88-0158

1/1

UNCLASSIFIED

F/G 12/1

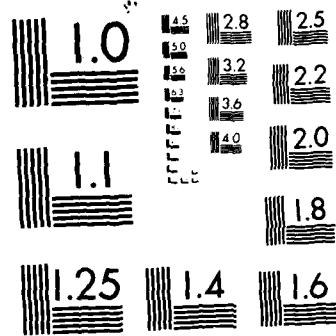
NL

END

FILMED

12/1

09/06



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963-A

AFOSR-TR- 33-0076

REPORT NO. 168

(3)

ADA125766

*A Vectorized
General Sparsity Solver*

D.A. CALAHAN

October 1, 1982



Sponsored by
Directorate of Mathematical and Information Sciences,
Air Force Office of Scientific Research,
under Grant No. 80-0758

Approved for public release;
distribution unlimited.

Systems Engineering Laboratory
The University of Michigan

DTIC FILE COPY

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFOSR-TR- 83-0076	2. GOVT ACCESSION NO. <i>A9-4125766</i>	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) A VECTORIZED GENERAL SPARSITY SOLVER	5. TYPE OF REPORT & PERIOD COVERED Interim	
7. AUTHOR(s) D. A. Calahan	6. PERFORMING ORG. REPORT NUMBER 168	
9. PERFORMING ORGANIZATION NAME AND ADDRESS University of Michigan Dept. of Elec. & Computer Engring. Ann Arbor, MI, 48109	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 61102F 2304/1A3	
11. CONTROLLING OFFICE NAME AND ADDRESS Air Force Office of Scientific Research Boeing AFB, Washington DC 20332	12. REPORT DATE October 1, 1982	
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	13. NUMBER OF PAGES 25	
	15. SECURITY CLASS. (of this report) UNCLASSIFIED	
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE	
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Sparse matrices Parallel processing Vector processing Linear algebra		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) Description and use of a Fortran general sparse solver, modified to run efficiently on a vector processor, is given. CRAY-1 performance in the analysis of 2D grids is presented.		

A Vectorized
General Sparsity
Solver

D. A. Calahan

Systems Engineering Laboratory
University of Michigan
Ann Arbor, Michigan 48109

October 1, 1982

SEL Report #168
Sponsored by
Directorate of Mathematical and Information Sciences
Air Force Office of Scientific Research,
under Grant 80-0158

Information

ABSTRACT

Description and use of a Fortran general sparse solver, modified to operate efficiently on a vector processor, is given. CRAY-1 performance of the solver in analysis of 2-D grids is presented.



A



I. INTRODUCTION

A. General Sparse Solvers

The performance of sparse equation solvers on vector processors is highly dependent on the machine instruction set and associated timings. The availability of indirect addressing instructions (gather/scatter) is a particularly important issue. When the processor has a hierarchical or other flexible memory organization, such as the CRAY-1, control of the data flow by assembly language coding may also become important.

Algorithmically, another set of considerations is introduced by the possibility of achieving high execution rates via equation re-ordering and recognition of favorable matrix structures. For example, blocking of matrices can reduce data traffic [7]; recognition of global sparsity patterns can also yield a vector solution [9].

B. Generic Vectorized Solver

In view of these opportunities for specialized algorithms, the value of a generic sparsity algorithm in a common high-level language for vector processors is in some question. For example, it is known that traditional general sparsity codes [2] execute poorly from a high level language on current vector processors, at rates less than 1/200 of maximum processor performance [10].

It is the viewpoint of this report that a generic vectorized solver has value in establishing a baseline performance against which the performance of more specialized solvers can be judged.

The following properties are proposed for such a solver.

- (a) Consistent with general solvers for scalar processors, the solver should accept column- or row-ordered symbolic matrix descriptions, plus permutation vectors describing row and column re-orderings.
- (b) The inner loop of the numeric solution should recognize opportunities to

exploit vector hardware.

(c) The solution may be of a two-phase symbolic/numeric nature, where a symbolic phase preprocesses the matrix structure and provides descriptors to a solution phase, for possible repeated numerical solutions with the same matrix structure [2].

II. The LU Map Approach

A. Introduction

The algorithm to be discussed is a variation of the "looped index" or "LU map" method of Chuang [1] and Gustavson [2] to vector processors. It was first discussed in detail in 1977 in [3] and its application to electrical circuit analysis given in [4]. The following discussion is taken from [3].

B. Symbolic Vectorization

1. Introduction

Given a matrix A and a right hand side B, it is proposed to perform a triangular factorization in the form

$$A = LU \quad (1)$$

where L and U are lower and upper triangular factors with elements l_{ij} and u_{ij} , respectively. Column ordered reduction is performed, with $l_{ii} = 1$. The forward and backward substitution has the form

$$LY = B \quad (2)$$

$$UX = Y \quad (3)$$

where X is the solution vector. (For descriptive clarity, pivoting is assumed down the main diagonal).

2. Scalar Model

In the work of Gustavson, the purpose of the symbolic phase was to determine the fill characteristics of A, i.e., the exact structure of L and U. This is a costly process that need be performed only once for a given matrix structure. To acquaint the reader with this approach, an example using Gustavson's "scalar" map is shown in Table 1. Special note should be taken of

- (1) the fill positions detected by the symbolic phase in the generation of the LU map;
- (2) the use of map indices in the numeric solution to extract information from the numeric arrays A, L, and U;
- (3) the use of an expanded current column (X array), requiring zeroing, expansion, and contraction in the loading and storing process;
- (4) the opportunities for the use of vector operations in the numeric solution, as evidenced by the indexed array operations marked "vector".

The following two sections are intended to give insight into the symbolic map generation by discussion of a proposed vectorized data structure and symbolic operations on it during the factorization process.

3. Vectorized List Data Structures

Consider a column of a sparse matrix having the non-zero row positions shown in Figure 1 (before fill). This structure would be described in a conventional ordered list as

$$31, 32, \dots, 36, 39, 42, 43, \dots, 47 \quad (4)$$

Such a list enumerating all row positions will be termed scalar storage. Clearly, the list can be shortened by identifying sets of contiguous positions (vectors) and retaining only the first and last row numbers, viz,

$$\begin{array}{ccccccccc}
 & 3 & 0 & 0 & 0 & 2 & & 3 & 0 & 0 & 0 & 0 & 2 \\
 & 0 & 4 & 2 & 1 & 0 & & 0 & 4 & 2 & 1 & 0 & 0 \\
 \Lambda = & 0 & 2 & 6 & 0 & 5 & \text{LU} = & 0 & 1/2 & 5 & -1/2 & 3 \\
 & 0 & 1 & 0 & 3 & 1 & & 0 & 1/4 & -1/10 & 27/10 & 13/10 \\
 & 2 & 0 & 3 & 1 & 5 & & 2/3 & 0 & 3/5 & 13/27 & 67/54 \\
 & & & & & & & & & & & \text{current} \\
 & & & & & & & & & & & \text{column} \\
 \text{matrix} & & & & & & \text{completely-factored matrix} & & & & &
 \end{array}$$

from user	A	(column-ordered numeric values of A matrix) 3,2,4,2,1,2,6,3,1,3,1,2,3,1,5
	JA	(JA(j) points to beginning of jth column of A in IA) 1,3,6,19,12,16
	IA	(column-ordered list of row numbers of A) 1,5,2,3,4,2,3,5,2,4,5,1,3,4,5
generated by symbolic	JL	(JL(j) points to beginning of jth column of L in IL) 1,2,4,6,7
	IL	(column-ordered list of row numbers of L) 5,13,4,4,5,5 fill
	JU	(JU(j) points to beginning of jth column of U in IU) 1,1,1,2,4,7
	IU	(column-ordered list of row numbers of U) 2,2,3,1,3,4 fill
	L	(column-ordered numeric values of L) 2/3,1/2,1/4,-1/10,3/5,
generated by numeric factorization	U	(column-ordered numeric values of U) 2,-1,-1,-1,-1
	DI	(ordered numeric values of diagonal) 3,4,5,

(a) Example up to factorization of fourth column

Table 1. Example of use of LU map in factorization

1. Zero expand current column (X array)
2. Load current column with fourth column of A

$X(2)=1$
 $X(4)=3$] vector
 $X(5)=1$]
 indices from IA

3. Factorize fourth column

$X(3)=X(3)-X(2)*L(2)=0-(1)(1/2)=-1/2$] vector
 $X(4)=X(4)-X(2)*L(3)=3-(1)(1/4)=11/4$] vector
 $X(4)=X(4)-X(3)*L(4)=11/4-(-1/2)(-1/10)=27/10$] vector
 $X(5)=X(5)-X(3)*L(5)=1-(-1/2)(3/5)=13/10$] vector
 indices from IL of previous columns

starting indices from JL

$DI(4)=1/X(4)=10/27$
 $X(5)=X(5)*DI(4)=13/27$

4. Store current column

$U(2)=X(2)$] vector
 $U(3)=X(3)$] vector
 $L(6)=X(5)$
 starting indices from JL, JU

indices from IL, IU of current column

(b) Steps in Factorization of Fourth Column

Table 1. Example of use of LU map in factorization

31,36,39,39,42,47

(5)

This form is natural to looping operations for a scalar processor, where pairs of numbers are directly usable as upper and lower loop indices. Alternatively, the initial row position and the vector length could be stored as

31,6,39,1,42,6

(6)

This form is favored by vector processors with hardware that counts down vector arithmetic operations to terminate a vector operation.

Another choice, preferred when a significant number of singleton (scalar) positions are present, represents a vector of length one with a minus sign prefixing the row number as

31,36,-39,42,47

(7)

This latter structure has been adopted in this report.

4. Vector Fills

The multiply-subtract inner loop associated with factorization can result in production of fills that must be detected in the symbolic phase. In Figure 1, the process of multiplying the k^{th} column of L (termed a *preceding* or *recalled* column) by $u_{k,r}$ and subtracting from the r^{th} column of L (termed the *current* column) is depicted. The zero-valued positions 37,38,40,41, which initially separate two vectors and a scalar, are filled by the dense vector (36,43) in the k^{th} column.

The symbolic phase produces the LU map by scanning the numbered pairs representing the vector structure of all the preceding columns and the current column to determine zero-valued regions of the latter covered by at least one of the former. These are the fill positions.

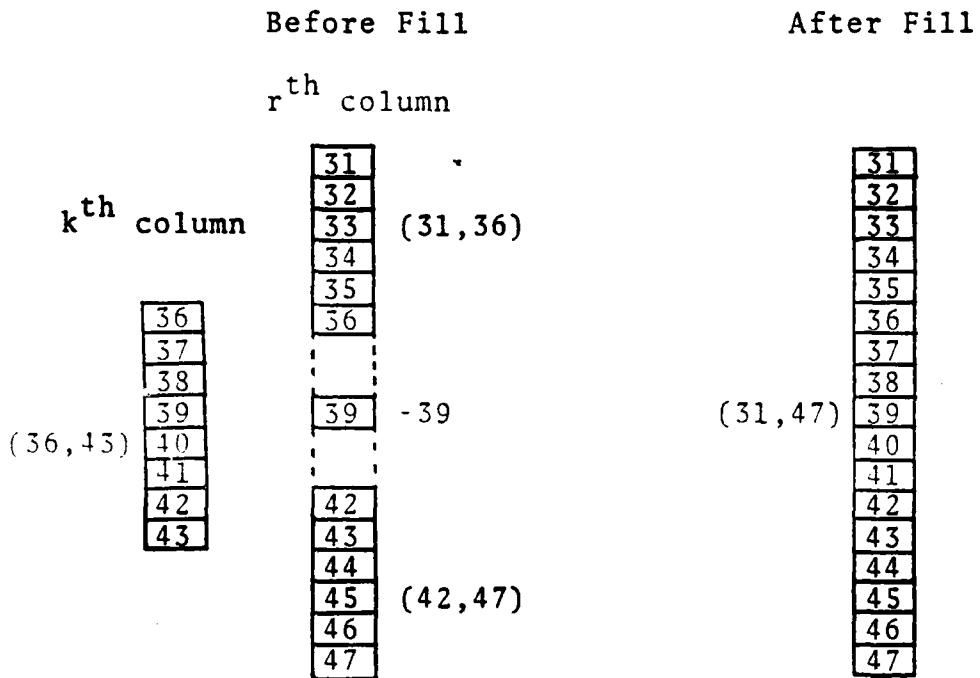


Figure 1. Example of vector fill, with data structure description (scalar indicated by - sign)

III. SOFTWARE DESCRIPTION

A. Symbolic preprocessing: Generation of compressed LU maps

CALL NEWFOR(N,IA,JA,IPC,IPR,IPRI,JL,JU,J,IL,IU,JVA,JVL,JVU,IX,IPOS,LENSC)

N* is number of equations.

IA(J)* on entry, IA(J) contains row number of jth column-ordered matrix element;
on exit, IA contains compressed vector-scalar format of same row number information.

JA(J)* points to first elements in IA of jth column;
JA(J) is changed by NEWFOR, as IA changes; JA(N+1)

*User supplied input data to subroutine.

IPC*	points to one beyond last element of IA. is column permutation vector.
IPR*	is row permutation vector.
IPRI	inverse row permutation vector.
JL(J) JU(J)	points to first element in IL and IU of jth column; dimensioned at least N+1.
IL(J) IU(J)	contains compressed vector-scalar row map of L and U
JVA(J) JVU(J) JVL(J)	points to first elements in numeric arrays A, U, and L of jth column; dimensioned at least N.
IX IPOS	scratch arrays of dimension N.
LENSC	is the maximum number (≥ 1) of contiguous non-zeros in a column that are processed in scalar mode.

B. Numeric solution

CALL VMNP (N,JA,IA,JVA,A,IU,IL,JU,JL,DI,U,L,X,JVU,JVL,IPC,IPRI)

(see above argument list for NEWFOR)

A*	array of numerical values of column-ordered matrix
DI	array of re-ordered diagonal elements of U.
U L	arrays of numerical values of column-ordered upper (U) and lower (L) triangular matrices; diagonal not included.
X	scratch array of dimension N.

C. Forward and back substitution.

CALI. VMBP(N,IU,IL,JU,JL,JVU,JVL,DI,U,L,B,X,IPC,IPR)

*User supplied input data to subroutine

(see above argument lists for lNEWFOR,VMNP)

B' array of numeric values of right hand side of entry,
and of solution on exit.

IV. PERFORMANCE

Three finite difference grids [5] illustrated in Figure 2 were solved using this code. The equations were ordered by alternate diagonals [6], which yields triangular LU factors of the form

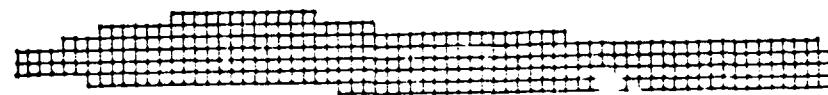
$$\begin{matrix} D_1 & U_{12} \\ L_{21} & U_{22} \end{matrix}$$

where D_1 is a diagonal matrix and U_{22} is profile matrix. Although U_{22} can be solved more efficiently [7][8], this code has the advantage of being in Fortran and simpler to use.

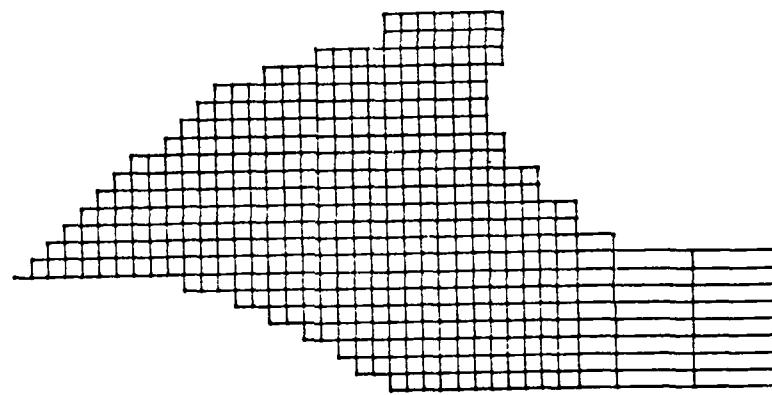
The performance on the CRAY-1 of the matrix factorizations step is depicted in Table 2. It should be noted that the execution rate is approximately proportional to the average vector length during solution. Of course, this is not true asymptotically, since the rate has a limiting value.

Problem	Equations	Time(msec)	MFLOPS	L
#1	391	20	1.59	4.2
#2	507	59	5.39	12.2
#3	2323	652	11.0	27.0

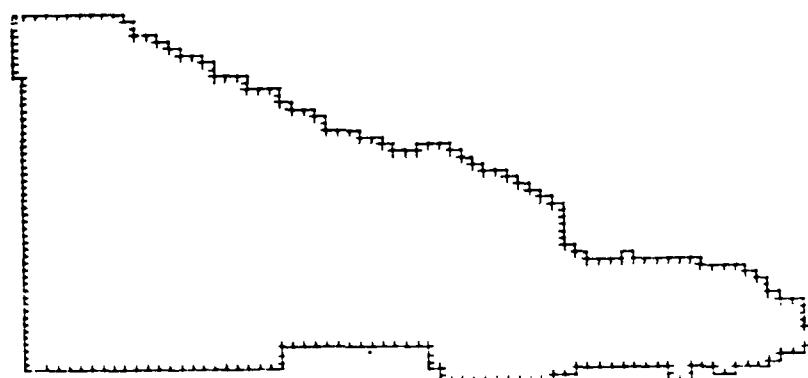
Table 2. CRAY-1 factorization performance summary for three grids of Figure 2;
L is average vector length.



(a) Problem #1, 8x69, 391 equations



(b) Problem #2, 23x37, 507 equations



(c) Problem #3, 55x72, 2323 equations

Figure 2. Irregular grids

References

- [1] Chang, A., "Application of Sparse Matrix Methods in Electric Power System Analysis," *Sparse Matrix Proceedings*, Willoughby, Editor, pp. 113-122.
- [2] Gustavson, F.G., "Some Basic Techniques for Solving Sparse Systems of Linear Equations," *Sparse Matrices and Their Applications*, Rose & Willoughby Editors, Plenum Press, 1972, pp. 41-52.
- [3] Calahan, D.A., P.G. Buning, and W.N. Joy, "Vectorized General Sparsity Algorithms with Backing Store," Report #96, Systems Engineering Laboratory, University of Michigan, January, 1977.
- [4] Calahan, D.A., and W.G. Ames, "Vector Processors: Models and Applications," IEEE Trans., vol. CAS-26, no. 9, September, 1979, pp. 715-726.
- [5] Woo, P.T., et al, "Application of Sparse Matrix Techniques to Reservoir Simulation," in *Sparse Matrix Computations*, ed. by J. R. Bunch and D. J. Rose, Academic Press, 1976, pp. 427-438.
- [6] Price, H.S., and K.H. Coates, "Direct Methods in Reservoir Simulation," Soc. Pet. Engrs. Jour., vol. 14, 1974, pp. 295-308.
- [7] Calahan, D.A., "High Performance Banded and Profile Solvers for the CRAY-1: The Unsymmetric Case," Report #160, Systems Engineering Laboratory, University of Michigan, February, 1982.
- [8] Calahan, D.A., "High Performance Banded and Profile Solvers for the CRAY-1: The Symmetric Case," Report #166, Systems Engineering Laboratory, University of Michigan, October, 1982.
- [9] Calahan, D. A., "Multi-level Vectorized Sparse Solution of LSI Circuits," Proc. IEEE Intl. Conf. on Circuits and Computers (1980), Rye, NY., pp. 976-979.
- [10] Calahan, D. A., "Vectorized Direct Solvers for 2- D Grids," Proc. 6th Symposium on Reservoir Simulation, New Orleans, Feb. 1-2, 1982, pp. 489-506.

Appendix A
Program Listing

```

1      C**** THIS IS A DRIVER PROGRAM FOR TESTING A PROGRAM NEWFOR
2      C THAT COMPRESSES A COLUMN-ORDERED SPARSE MATRIX DESCRIPTION.
3      C AND PROGRAMS VMPN AND VMBP THAT FACTOR AND SOLVE THE
4      C MATRIX (RESPECTIVELY). NEWFOR NEED BE INVOKED ONLY ONCE
5      C FOR MULTIPLE SOLUTIONS WITH VMPN AND VMBP
6      C**** SCALAR/VECTOR GENERAL SPARSE SOLVER
7      C**** DOCUMENTATION IN "VECTORIZED GENERAL SPARSITY ALGORITHMS
8      C**** WITH BACKING STORE," D. A. CALAHAN, P. G. BUNING AND W. N. JOY
9      C**** REPORT #96, U. OF MICHIGAN, JANUARY 1977; AND IN "USERS
10     C**** MANUAL FOR VECTORIZED GENERAL SPARSE SOLVER," BY D. A. CALAHAN
11     C**** OCTOBER 1982
12    C
13    C
14    C**** THE FOLLOWING ARRAYS HAVE A DIMENSION .GE. N
15    REAL A,B,DI,L,U,X,SUMR,SUMC
16    DIMENSION IPR(2325),IPC(2325),IPRI(2325),IX(2325)
17    8,JVA(2325),JVL(2325),JVR(2325),SUMC(2325),B(2325)
18    8,X(2325),DI(2325)
19    C**** THE FOLLOWING ARRAYS HAVE A DIMENSION .GE. N+1
20    C DIMENSION IPOS(2326),JA(2326),JU(2326),JL(2326)
21    C**** THE FOLLOWING ARRAYS HAVE DIMENSIONS .GE. TO THE NUMBER
22    C OF NON-ZEROS IN THE MATRIX
23    DIMENSION A(1200),IA(12000)
24    C**** THE FOLLOWING ARRAYS HAVE DIMENSIONS .GE. TO THE NUMBER
25    C OF NON-ZEROS OF L AND U; IU AND IL ARE COMPRESSED AND
26    C MAY REQUIRE MUCH LESS THAN THIS PESSIMISTIC ESTIMATE
27    C DIMENSION TU(6700),IL(6700)
28    COMMON /EX2/U(86000)
29    COMMON /EX1/L(86000)
30    READ(5,88)N
31    NP=N+1
32    READ(5,88)(JA(J),J=1,NP1)
33    NA=JA(NP1)-1
34    READ(5,88)(IA(J),J=1,NA)
35    FORMAT(16I5)
36    DO 89 J=1,N
37    IPR(J)=J
38    89  IPC(J)=J
39    C18  CALL RANGEN(JA,IA,N,IPR,IPC)
40    C  NP1=N+1
41    C  WRITE(6,17)(JA(J),J=1,NP1)
42    C  NA=JA(NP1)-1
43    C  WRITE(6,17)(IA(J),J=1,NA)
44    C17  FORMAT(20I3)
45    LENSC=2
46    CALL FORM(A,B,IA,JA,IPR,IPC,SUMR,SUMC,N,NA)
47    CALL NEWFOR(IA,JA,IPC,IPR,IPRI,JL,JU,IL,IU,JVA,JVL)
48    1 JVU,IX,IPOS,N,LENSC)
49    CALL VMPN(N,JA,IA,JVA,A,IU,IL,JU,JL,DI,U,L,X,JVU,JVL,IPC,IPRI)
50    CALL VMBP(N,IU,IL,JU,JVL,DI,U,L,B,X,IPC,IPR)
51    WRITE(7,77)(B(J),J=1,N)
52    FORMAT(5E12.4)
53    GO TO 18
54    END

```

+-----+
 | DATE : 09-28-82, 11:00 OWNER : SMXA FILE : SPARF |
 +-----+

```

55      C**** THIS SUBROUTINE COMPRESSES AN ARRAY OF 0'S AND 1'S (IPOS)
56      C IN VECTOR-SCALAR FORM
57      C
58      SUBROUTINE ILU(IU,JU,IPOS,IV,NU,NUV,ISTR,IE,LEN,SC)
59      DIMENSION IV(1),IU(1),JU(1),IPOS(1)
60      IV(J)=NUV+1
61      JU(J)=NU+1
62      IF(IE,GT,0)GO TO 7
63      I=ISTR
64      IPL=0
65      IPC=IPOS(1)
66      IP0S(I)=0
67      I=I+1
68      IPN=IPOS(1)
69      IF(IPC,NE,0)GO TO 2
70      IF(I,GT,IE)GO TO 7
71      IPC=IPC
72      IPN=IPN
73      I=I+1
74      IPN=IPOS(1)
75      GO TO 3
76      IP0S(I-1)=0
77      IF(IPL,NE,0)GO TO 4
78      IF(IPN,NE,0)GO TO 6
79      C**** SCALAR
80      NU=NU+1
81      NUV=NUV+1
82      IU(NU)=-(I-1)
83      GO TO 5
84      C**** START OR END OF VECTOR
85      IF(IPN,NE,0)GO TO 5
86      LEN=I-IU(NU)
87      NUV=NUV+LEN
88      IF(LEN,GT,LEN,SC)GO TO 6
89      II=-(I-1)+LEN
90      NU=NU-1
91      DO 8 L=1,LEN
92      NU=NU+1
93      IU(NU)=II-L
94      GO TO 5
95      NU=NU+1
96      IU(NU)=(I-1)
97      GO TO 5
98      7 RETURN
99

```

+-----+
 | PAGE 2 |
 +-----+

1
 2
 3
 4
 5
 6
 7
 8
 9
 10
 11
 12
 13
 14
 15
 16
 17
 18
 19
 20
 21
 22
 23
 24
 25
 26
 27
 28
 29
 30
 31
 32
 33
 34
 35
 36
 37
 38
 39
 40

```

100      SUBROUTINE RANGEN(JA,IA,N,IPR,IPC)          1
101      C***  GENERATES RANDOMLY-POSITIONED TEST MATRICES
102      DIMENSION IPR(1),IPC(1),IA(1),JA(1)          2
103      C***  AROW IS AVERAGE NUMBER OF NON-ZEROS PER ROW
104      C***  N IS THE NUMBER OF EQUATIONS
105      READ(5,3)AROW,N
106      FORMAT(F10.0,16)                            3
107      NA=0                                         4
108      NNN=999                                      5
109      AN=N                                         6
110      TOL=(AROW-1)/AN                           7
111      JA(1)=1                                     8
112      DO 33 J=1,N                                9
113      DO 1 K=1,N                                 10
114      IF (J.EQ.K)GO TO 5                         11
115      IF (RANF(NNN).GT.TOL)GO TO 1             12
116      NA=NA+1                                     13
117      IA(NA)=K                                    14
118      1    CONTINUE                                15
119      33    JA(J+1)=NA+1                         16
120      DO 37 J=1,N                                17
121      IPR(J)=J                                    18
122      IPC(J)=J                                    19
123      RETURN                                     20
124      END                                         21
                                         22

```

```

DATE: 09-28-82, 11:00 OWNER: SMXA FILE: SPARF

125      SUBROUTINE VMNP(N,JA,IA,JVA,A,IU,IL,JU,JL,DI,U,X,JVU,JVL,
126      C IPC,IPRI)
127      C*****+
128      C*      V M N P - SPARSE LU FACTORIZATION (NUMERIC) *
129      C*      ****
130      C*      ****
131      C*      ****
132      C*      THIS SUBROUTINE PERFORMS THE NUMERIC LU FACTORIZATION *
133      C*      A GENERAL NON-SINGULAR SPARSE MATRIX. IT OPERATES ON *
134      C*      ****
135      C*      ****
136      C*      ****
137      C*      ****
138      C*      ****
139      C*      INTEGER IPC,IPRI,IA,JU,JL
140      C      REAL L
141      C      IPC - COLUMN PIVOT PERMUTATION VECTOR
142      C      IPRI - INVERSE ROW PIVOT PERMUTATION VECTOR
143      C
144      C      DIMENSION IPC(1),IPRI(1)
145      C      C - NUMBER OF COLUMNS IN THE MATRIX.
146      C
147      C      DIMENSION JA(1),IA(1),JVA(1),A(1)
148      C
149      C      DIMENSION JU(1),JL(1),IU(1),IL(1),JVU(1),JVL(1)
150      C
151      C      JA(1) - START OF COLUMN POSITION DESCRIPTORS IN IA FOR COLUMN 1.
152      C      IA - IF NEGATIVE, IABS IS STARTING ROW OF A SINGLE ELEMENT.
153      C      - IF POSITIVE, SHARTING COULMN FOR A VECTOR AND THE
154      C      NEXT ELEMENT OF IA IS ENDING ROW FOR THIS VECTOR.
155      C      JVA - POINTER TO FIRST COLUMN IA MATRIX VALUE (IN A.)
156      C      A - ABOVE GROUP OF ARRAYS DESCRIBE INFO IN THIS ARRAY.
157      C
158      C      DIMENSION JU(1),JL(1),IU(1),IL(1),JVU(1),JVL(1)
159      C
160      C      JU(I) - START OF ROW POSITION DESCRIPTORS IN JU FOR COL I
161      C      JL(I) - START OF ROW POSITION DESCRIPTORS IN JL FOR COL I
162      C      JU - IF NEGATIVE, IABS IS ROW INDEX OF SINGLE ELEMENT.
163      C      - IF POSITIVE, STARTING ROW FOR A VECTOR, AND THE
164      C      NEXT ELEMENT OF JU IS ENDING ROW FOR THIS VECTOR.
165      C      JL - IF NEGATIVE, IABS IS ROW INDEX OF SINGLE ELEMENT.
166      C      - IF POSITIVE, STARTING ROW FOR A VECTOR, AND THE
167      C      NEXT ELEMENT OF JL IS ENDING ROW FOR THIS VECTOR.
168      C      JVU - INDEX OF FIRST COLUMN I J MATRIX VALUE.
169      C      JVL - INDEX OF FIRST COLUMN I L MATRIX VALUE.
170      C
171      C      DIMENSION U(1),L(1)
172      C      C U - VECTOR OF UPPER TRIANGULAR NUMERICAL VALUES.
173      C      C L - VECTOR OF LOWER TRIANGULAR NUMERICAL VALUES.
174      C
175      C      DIMENSION DI(1), X(1)
176      C
177      C      C DI - VECTOR OF INVERSE DIAGONAL ELEMENT VALUES
178      C      C X - WORK VECTOR OF LENGTH N.
179      C
180      C
181      C
182      C

```

C STOP 27201 - NUMERICAL VALUE OF PIVOT IS APPROXIMATELY 0.

C ****

C INITIALIZE POINTERS:

C JUU IS INDEX INTO L MATRIX VALUE VECTOR.

C (USED TO STORE VALUES IN L AND U CALCULATION LOOPS.)

C JAA IS INDEX INTO A MATRIX VALUE VECTOR.

C (USED TO RETRIEVE VALUES FROM A MATRIX FOR EACH COLUMN.)

C ****

C JUU= 0
JLL= 0
ILBOT= 0
IABOT= 0
IUBOT= 0
DO 17 I=1,N

17 X(1)=0.

C ***** C LOOP TO CALCULATE NUMERICAL VALUES FOR COLUMN I OF L AND L

C ****

DO 80 I= 1,N

204 C GET POINTERS FOR THIS COLUMN IN L AND L:

205 C IUTOP IS INDEX TO TOP OF ROW DESCRIPTORS FOR THIS COLUMN IN U.

206 C IUBOT IS INDEX OF BOTTOM OF ROW DESCRIPTORS FOR THIS COLUMN IN U.

207 C ILTOP IS INDEX TO TOP OF ROW DESCRIPTORS FOR THIS COLUMN IN L.

208 C ILBOT IS INDEX TO BOTTOM OF ROW DESCRIPTORS FOR THIS COLUMN IN L.

209 C IUTOP IS INDEX TO TOP OF ROW DESCRIPTORS FOR THIS COLUMN IN L.

210 C IUBOT= IUBOT+1
IUTOP= IUBOT+1
IUBOT= JU(I+1)-1211 C ILTOP= ILBOT+1
ILTOP= ILBOT+1
ILBOT= JL(I+1)-1

212 C ****

213 C GET POINTER TO DATA VALUE DESCRIPTORS FOR A:

214 C IATOP IS INDEX TO TOP OF ROW INDICES FOR THIS COLUMN.

215 C IABOT IS INDEX TO BOTTOM OF ROW INDICES FOR THIS COLUMN.

216 C **** NOTE *** WE ARE PROCESSING COLUMN IPC(1) OF A ***

217 C ****

218 C ****

219 C ****

220 C ****

221 C ****

222 C ****

223 C ****

224 C INITIALIZE WORKSPACE X WITH ZEROS AT RESULTANT POSITIONS

225 C **** COULD APPROXIMATE - ZERO LOWEST TO HIGHEST POSITION

226 C ****

227 C IF THERE ARE NO NONZERO POSITIONS IN L THEN THE ONLY RESULTANT

228 C POSITIONS WILL BE THE ELEMENTS IN L. SINCE THESE WILL BE GIVEN

229 C INITIAL VALUES IN THE LOOP BEGINNING AT #31 BELOW, WE DONT NEED

230 C TO ZERO ANY POSITIONS IN THIS CASE.

231 C IF (IUTOP.GT.IUBOT) GOTO 30

232 C ZEROS IN X AT POSITIONS OF COLUMNS OF JU

233 C K= IUTOP

234 C10 JUI= IU(K)

235 C IF(JUI.LT.0) GOTO 12

236 C K= K+1

237 C JUE= IU(K)

238 C DO 11 J= JUI , JUE

239 C X(J)= 0.

240 C11

241 C GOTO 13
 242 C12 X(-JUJ)= 0.
 243 C K= K+1
 244 C IF (K.LE.IUBOT) GOTO 10
 245 C ZEROS IN X AT POSITIONS OF COLUMNS OF JL
 246 C IF (ILTOP.GT.IUBOT) GOTO 30
 247 C K= ILTOP
 248 C JL1= IL(K)
 249 C IF (JL1.LT.0) GOTO 22
 250 C K= K+1
 251 C JL1= IL(K)
 252 C DO 21 J= JL1,JLE
 253 C X(J)= 0.
 254 C GOTO 23
 255 C22 X(-JUJ)= 0.
 256 C K= K+1
 257 C IF (K.LE.IUBOT) GOTO 20
 258 C COPY COL'MN IPC(I) OF A INTO X
 259 C JN11NU
 30 JAA= JVA(IPC(I))-1
 260 K= IATOP
 261 31 JAP= IA(K)
 262 C IF (JAP.LT.0) GOTO 33
 263 C JN11NU
 264 C K= K+1
 265 C JAL=IA(K)-JAP+1
 266 C DO 32 J= 1,JAL
 267 C X(IPRI(JAP+J-1))= A(JAA+J)
 268 C JAA= JAA+JAL
 269 C GOTO 34
 270 33 JAA= JAA+1
 271 C X(IPRI(-JAP))= A(JAA)
 272 C K= K+1
 273 C IF (K.LE.IABOT) GOTO 31
 274 C
 275 C*****
 276 C CALCULATE ENTRIES IN COLUMN 1 OF UPPER TRIANGULAR MATRIX U
 277 C*****
 278 C NOTHING TO DO IF COLUMN 1 OF U CONTAINS ONLY THE DIAGONAL ELEMENT
 279 C IF (IUTOP.GT.IUBOT) GOTO 60
 280 C LOOP ON EACH DESCRIPTOR IN JU MATRIX FOR THIS COLUMN IN U.
 281 C J= IUTOP
 282 40 JUJ= IU(J)
 283 C GET NEXT ELEMENT FROM COLUMN 1
 284 C IF (JUJ.GT.0) GOTO 41
 285 C JUJ= -JUJ
 286 C JUJL= JUJ
 287 C GOTO 42
 288 C J= J+1
 289 41 JUJL= IU(J)
 290 C LOOP ON EACH POSITION IN THE ELEMENT
 42 C DO 54 LL= JUJ,JUJL
 291 C XJI= X(LL)
 292 C X(LL)= 0.
 293 C U(JUJ+LL-JUJ+1)= XJI
 294 C LOCATE ROW DESCRIPTOR INDICES FOR COLUMN LL IN U
 295 C JTOP= JL(LL)
 296 C JBOT= JL(LL+1)-1
 297 47
 298 48
 49
 50
 51
 52

L(J,L)= XD*X(-J,L)

357 X(-J,L)=0

358 J= J+1

73 IF (J.LE.1LBOT) GOTO 70

360 C***** END OF COMPUTATION OF COLUMN I OF L *****

361 80 CONTINUE

362 RETURN

363 90 STOP 27201

364 END

365

<PAGE 8>

DATE: 09-28-82, 11:00	OWNER: SMXA	FILE: SPARF
		ISN

```

366      SUBROUTINE VMBP(N,IU,IL,JU,JL,JVU,JVL,DI,U,L,B,X,IPC,IPR)      1
367      C*      V M B P - SPARSE FORWARD AND BACK SUBSTITUTION      *
368      C*      ****
369      C*      ****
370      C*      ****
371      C*      ****
372      C*      ****
373      C*      THIS SUBROUTINE SOLVES THE MATRIX EQUATION A*X=B GIVEN      *
374      C*      THE LU FACTORED A MATRIX (FROM VMNSP) AND THE VECTOR B      *
375      C*      ****
376      C*      **** ARBITRARY PIVOTING ORDER VERSION ****      *
377      C*      ****
378      C*      ****
379      C*      INTEGER JU, JL, IPC, IPR      2
380      C*      REAL L      3
381      C      N - NUMBER OF COLUMNS IN THE MATRIX.      4
382      C      DIMENSION JU(1), JL(1), IU(1), IL(1), JVU(1), JVL(1)
383      C
384      C      JU(I) - START OF ROW POSITION DESCRIPTORS IN JU FOR COL I
385      C      JL(I) - START OF ROW POSITION DESCRIPTORS IN JL FOR COL I
386      C      IU - IF NEGATIVE, IABS IS ROW INDEX OF SINGLE ELEMENT.
387      C      - IF POSITIVE, STARTING ROW FOR A VECTOR, AND THE
388      C      NEXT ELEMENT OF JU IS ENDING ROW FOR THIS VECTOR.
389      C      - IF NEGATIVE, IABS IS ROW INDEX OF SINGLE ELEMENT.
390      C      - IF POSITIVE, STARTING ROW FOR A VECTOR, AND THE
391      C      NEXT ELEMENT OF JL IS ENDING ROW FOR THIS VECTOR.
392      C      - IF POSITIVE, STARTING ROW FOR A VECTOR, AND THE
393      C      INDEX OF FIRST COLUMN I J MATRIX VALUE.
394      C      JVU - INDEX OF FIRST COLUMN I J MATRIX VALUE.
395      C      JVJL - INDEX OF FIRST COLUMN I L MATRIX VALUE.
396      C
397      C      DIMENSION U(1), L(1), B(1), X(1), DI(1)      5
398      C      U - VECTOR OF UPPER TRIANGULAR NUMERICAL VALUES.
399      C      L - VECTOR OF LOWER TRIANGULAR NUMERICAL VALUES.
400      C      X - SCRATCH VECTOR OF LENGTH N.
401      C      B - VECTOR OF RIGHT HAND SIDE VALUES.
402      C      DI - VECTOR OF INVERSE DIAGONAL ELEMENT VALUES.
403      C      - RETURNED SOLUTION VECTOR FOR EQUATION A*X=B
404      C
405      C      DIMENSION IPC(1), IPR(1)      6
406      C
407      C      IPC - COLUMN PIVOT PERMUTATION VECTOR.
408      C      IPR - ROW PIVOT PERMUTATION VECTOR.
409      C
410      C      NM1= N-1      7
411      C
412      C
413      C      DO 5 J= 1,N      8
414      C      IPRX= IPR(J)      9
415      C      X(J)= B(IPRX)      10
416      C
417      C      ****
418      C      FORWARD SUBSTITUTION      11
419      C
420      C      DO 30 I= 1,NM1      12
421      C      ITOP= JL(I)      13
422      C      IBOT= JL(I+1)-1      14
423      C      JV1= JVJL(I)-1

```



```

472
473      SUBROUTINE VSORT(A,N)
474      C  PARTITION SORTING ALGORITHM
475      C  REFERENCE COLLECTED ALGORITHMS OF THE ACM - 63,64,65
476
477      INTEGER A(1)
478      DIMENSION IHIGH(32),ILOW(32)
479
480      C INITIALIZE
481      NSEGS= 1
482      IH= N
483      C IF NO ELEMENTS IN THIS SEGMENT DO NOTHING
484      10     IF (IL.GE.IH) GOTO 70
485      C CHOOSE ISEP (SEPARATION ENTRY):
486      C MAKE A(IL) <= A((IL+IH)/2) <= A(IH) BY INTERCHANGE
487      C SET ISEP= A((IL+IH)/2)
488      20     ISEP= (IH+IL)/2
489      ISEP= A(ISEPX)
490      C IXL IS LOWER SEGMENT INDEX (CURRENT)
491      IXL= IL
492      C MAKE A(IL) <= A(ISEPX)
493      IF (A(IL).LE.ISEP) GOTO 30
494      A(ISEPX)= A(IL)
495      A(IL)= ISEP
496      ISEP= A(ISEPX)
497      C IXH IS HIGHEST SEGMENT INDEX (CURRENT)
498      IXH= IH
499      C MAKE A(IH) >= A(ISEPX)
500      IF (A(IH).GE.ISEP) GOTO 50
501      A(ISEPX)= A(IH)
502      A(IH)= ISEP
503      ISEP= A(ISEPX)
504      C MAKE A(IL) <= A(ISEPX)
505      IF (A(IL).LE.ISEP) GOTO 50
506      A(ISEPX)= A(IL)
507      A(IL)= ISEP
508      ISEP= A(ISEPX)
509      GOTO 50
510
511      C EXCHANGE LOW PART ENTRY WHICH IS GREATER THAN SEPARATOR WITH HIGH
512      C PART ENTRY WHICH IS LESS THAN OR EQUAL TO THE SEPARATOR VALUE.
513      40     ITT= A(IXH)
514      A(IXH)= A(IXL)
515      A(IXL)= ITT
516      C MOVE DOWN UPPER SEGMENT AS FAR AS WE CAN
517      50     IXH= IXH-1
518      C MOVE UP LOWER SEGMENT AS FAR AS WE CAN
519      51     ITT= IXL+1
520      C NOTHING TO DO IF BOTH SEGMENTS HAVE AT MOST ONE ENTRY IN COMMON
521      C IF BOTH SEGMENTS OVERLAP THEN THEY ARE SEPARATED
522      C IN THIS CASE CONTINUE WITH SHORTER SEGMENT, STORING THE LONGER
523      C IF (IXH-IL.LE.IH-IXL) GOTO 60
524      C LOWER SEGMENT LONGER, CONTINUE WITH UPPER AFTER SAVING LOWER
525      IL= IXL
526      ILW(NSEGS)= IL
527      IHIGH(NSEGS)= IXH
528
529

```

```

DATE : 09-28-82, 11:00 OWNER : SMXA FILE : SPARF
ISN

530      NSEGS= NSEGS+1
531      GOTO 80
532      C UPPER SEGMENT LONGER.  CONTIN WITH LOWER AFTER SAVING UPPER
533      60      ILOW(NSEGS)= IXL
534      IHIGH(NSEGS)= IH
535      IH= IXH
536      NSEGS= NSEGS+1
537      GOTO 80
538      C GET ANOTHER SEGMENT FOR PROCESSING IF THERE ARE ANY MORE
539      70      NSEGS= NSEGS-1
540      IF (NSEGS.EQ.0) RETURN
541      IL= ILOW(NSEGS)
542      IH= IHIGH(NSEGS)
543      C CONTINUE TO SEGMENT AS LONG AS LENGTH IS GREATER THAN 11
544      80      IF (IH-IL GE. 11) GOTO 20
545      IF (IL.EQ. 1) GOTO 10
546      GOTO 91
547      C SORT ELEMENTS WITHIN SEGMENT BY INTERCHANGE OF ADJACENT PAIRS
548      90      IL= IL+1
549      91      IF (IL.EQ.IH) GOTO 70
550      ISEP= A(IL+1)
551      IF (A(IL).LE.ISEP) GOTO 90
552      IXL= IL
553      100     A(IL+1)= A(IXL)
554      IXL= IXL-1
555      IF (ISEP.LT.A(IXL)) GOTO 100
556      A(IL+1)= ISEP
557      GOTO 90
558      END

```

```

559      SUBROUTINE FORM(A,B,IA,JA,IPR,IPC,SUMR,SUMC,N,NA)      1
560      DIMENSION A(1),B(1),IA(1),JA(1),IPR(1),IPC(1),SUMR(1),SUMC(1) 2
561      DO 12 I=1,NA                                              3
562      A(I)=0.                                              4
563      C*** UNIFORMLY-DISTRIBUTED NEGATIVE OFF-DIAGONAL VALUES 5
564      NNN=999                                              5
565      DO 29 J=1,NA                                              6
566      A(J)=-RANF (NNN)                                         6
567      DO 103 J=1,N                                              7
568      SUMR(J)=0.                                              8
569      SUMC(J)=0.                                              9
570      B(J)=0.                                              10
571      C*** FORMULATE EQUATIONS SO SOLUTION IS X(I) = I + 1 11
572      DO 37 I=1,N                                              12
573      I1=JA(I)                                              13
574      I2=JA(I+1)-1                                         14
575      DO 4 J=I1,I2                                         15
576      ICOL=IA(J)                                              16
577      SUMC(I)=SUMC(I)-A(U)                                         16
578      SUMR(ICOL)=SUMR(ICOL)-A(J)                                         17
579      B(ICOL)=B(ICOL)+A(U)**(I+1)                                         18
580      CONTINUE                                              19
581      C*** FIND PIVOTS AND FORCE DOMINANCE 20
582      DO 33 II=1,N                                              21
583      I=IPC(II)                                              22
584      I1=JA(I)                                              23
585      I2=JA(I+1)-1                                         24
586      DO 34 J=I1,I2                                         25
587      ICOL=IA(J)                                              26
588      IF(ICOL.NE.IPR(II))GO TO 34                           27
589      B(ICOL)=B(ICOL)-A(U)*(I+1)                                         28
590      A(U)=.0+1.1D0*AMAX((SUMC(I)+A(J),SUMR(ICOL)+A(J))) 29
591      B(ICOL)=B(ICOL)+A(U)*(I+1)                                         30
592      GO TO 33                                              31
593      34  CONTINUE                                              32
594      33  CONTINUE                                              33
595      RETURN                                              34
596      END                                              35

```

```

597      THIS SUBROUTINE GENERATES COMPRESSED MAPS OF A,U, AND L
598      SUBROUTINE NEWFOR(IA,JA,IPC,IPR,IPRI,JL,JU,IU,JVA,JVL,
1     JUJU,IX,IPOS,N,LWNSC)          1
599      DIMENSION IA(1),JA(1),IPC(1),IPR(1),JL(1),JU(1),IU(1),
2     &JVA(1),JVL(1),JUJU(1),IX(1),IPOS(1)          2
600      C**** DETERMINE INVERSE ROW PERMUTATION VECTOR
601      602      DO 104 J=1,N          3
602      IPRI(IPR(J))=J          4
603      NP1=N+1          5
604      DO 130 J=1,NP1          6
605      IPOS(J)=0          7
606      C**** DETERMINE FILL OF REORDERED MATRIX AND GENERATE COLUMN-ORDERED
607      C      LIST OF L AND U STRUCTURE IN VECTOR-SCALAR FORM          8
608      C
609      NU=0          9
610      NL=0
611      NU=0
612      NLV=0
613      DO 1 J=1,N          10
614      IP=IPC(J)          11
615      I1=JA(IP)          12
616      I2=JA(IP+1)-1          13
617      I3=0          14
618      DO 2 I=1,12          15
619      I3=I3+1          16
620      2 IX(I3)=IPRI(IA(I))          17
621      CALL VSORT(IX,I3)          18
622      IMIN=IX(1)          19
623      DO 10 I=1,13          20
624      IPOS(IX(I))=I          21
625      IMAX=IX(I3)          22
626      L1=0          23
627      L2=0          24
628      JM1=J-1          25
629      IF (IMIN.GT.JM1) GO TO 11          26
630      DO 3 I=IMIN,JM1          27
631      IF (IPOS(I).EQ.0) GO TO 3          28
632      K1=JL(I)          29
633      K2=JL(I+1)-1          30
634      IF (K1.GT.K2) GO TO 3          31
635      K=K1-1          32
636      L1=IL(K)          33
637      IF (L1.LT.0) GO TO 7          34
638      K=K+1          35
639      L1=IL(K)          36
640      IF (L1.LT.0) GO TO 3          37
641      L2=IL(K)          38
642      DO 6 LL=L1,L2          39
643      IPOS(LL)=1          40
644      GO TO 8          41
645      7 IPOS(-L1)=1          42
646      8 IF (K.LT.K2) GO TO 5          43
647      IMAX=MAX0(IMAX,L2)          44
648      IMAX=MAX0(IMAX,-L1)          45
649      3 CONTINUE          46
650      11
651      IPOS(J)=0          47
652      CALL ILU(IU,JU,IPOS,JUJU,NU,NUV,IMIN,J-1,J.LENSC)          48
653      JU(J+1)=NU+1          49
654      CALL ILU(IU,JL,IPOS,JVL,NL,NLV,J+1,IMAX,J.LENSC)          50
655          51
656          52

```

| DATE:09-28-82, 11:00 OWNER:SMXA FILE:SPARF | ISN

```

655      JL(J+1)=NL+1
656      CONTINUE
657      C****  COMPRESS MATRIX (A) STRUCTURE INTO VECTOR-SCALR FORM
658      NA=0
659      NAV=0
660      DO 109 J=1,N
661      I1=JA(J)
662      I2=JA(J+1)-1
663      IMAX=0
664      IMIN=10000
665      DO 108 I=I1,I2
666      IAX=IA(I)
667      IPOS(IAX)=1
668      IMIN=MINO(IAX,IMIN)
669      IMAX=MAXO(IAX,IMAX)
670      CALL ILU1A(JA,IPOS,JVA,NA,NAV,IMIN,IMAX,J,LENSC)
671      JA(NP1)-NA+1
672      FORMAT(2013)
673      RETURN
674      END

```

